Applying Statistical Methods in Knowledge Management of a Multiagent System

Michal Košinár – Ondřej Kohut

Abstract: Multi-agent system is a system of autonomous, intelligent but resource-bounded agents. Particular agents have to be able to make decisions on their own, based on the activities of other agents and events within the system as well as in its environment. To this end agents make use of their own internal knowledge base which serves them as a memory. In this paper we focus on the design and management of such a knowledge base. After a brief description of some classical fundamental approaches to the knowledge base management, we propose an improvement based on the application of statistical methods. We focus in particular on the optimization of the process.

Keywords: knowledge, rules, facts, TIL, TIL-Script, Transparent Intensional Logic, PROLOG, Multiagent Systems, content language, FIPA, a priori probability, conditional probability, Bayesians probability.

1 Introduction

The technology of multi-agent systems is rather a new technology which is still rapidly developing. One of the main problems a multiagent system must deal with is communication and reasoning of agents. This problem gives rise to another one, namely the need for agents' internal knowledge base, because agents' reasoning and communication is based on their knowledge. In order to meet agents' needs for their own knowledge, we must take care of the whole process, beginning with requirements analysis continued by knowledge

design and knowledge base realization. In this paper we describe our approach to the design (data model) and partly also realization of the designed model (knowledge management).

The paper is organized as follows. Section 2 describes the basic principles applied in agents' communication and interaction with environment and also content languages. In Section 3 we discuss Knowledge Bases with ontologies. The main Section 4 is a proposal of a knowledge base model as well as a conceptual/mathematical model of knowledge management, in particular knowledge deleting. Concluding remarks are contained in Section 5.

2 Multi-agent Systems and Communication

Technologies based on autonomous agents are relatively new and promising. Numerous applications of multi-agent technology can be found in the area of artificial intelligence and large computer systems. A road-map of this approach is presented in Luck – McBurney - Shehory – Willmott (2005). In this paper we do not intend to deal with multi-agent systems (MAS) in general. Instead, we focus on communication in MAS, Knowledge Bases and Knowledge Management.

Basic standards for MAS are given by FIPA (The Foundation for Intelligent Physical Agents, see 2000a and 2000b). According to these standards basic unit of communication is a message. It can be of an arbitrary form, but it is supposed to have a structure containing several attributes. *Content* of a message is one of these attributes.

From the point of view of communication logic, the most important attributes are:

Performative denotes a type of the message – its communicative act. Basic performatives are: *Query, Inform and Request*.

Content is the semantic core of the message. It can be encoded in any suitable language.

Ontology is a vocabulary of domain specific terms. These (and only these) terms can be used in the content of the message.

2.1 Agent and Environment Interaction

In order to introduce communication based on agents' knowledge, we are going to describe agents' reactions to the events in their environment, and interaction with the environment in general.

Figure 1 illustrates agents' interaction with the environment.

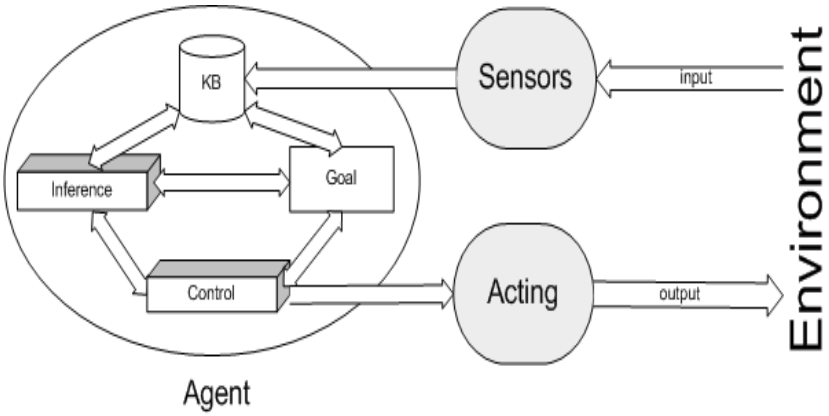

Figure 1. Behavior of agents in a real environment

Agents are autonomous, rational and goal-oriented. In order to be able to actively react on the events in their environment, they have to be equipped with:

- *Sensors* – "Ears", "Eyes"
- *Acting parts* – "Mouth" for communication, "Limbs" for an active reaction (movement, etc.)
- *Knowledge-Base* based on ontologies. This part serves as an agents' memory that makes it possible to store perceived or learnt facts, explicit and inferred knowledge, as well as general rules. (At least a minimal) ontology is needed to be shared with other agents, so that the agents' understand each other.
- *Inference engine* that is based on Description Logic, FIPA SL, Prolog, etc. In our system we vote for the *TIL-Script language* due to its great expressive power and *procedural semantics*, which make the language apt for natural-language analysis and

communication. In principal, *TIL-Script* is a computational variant of Transparent Intensional Logic (TIL). The language is still under development. It is designed as a functional, declarative programming language that will be of the same expressive power as the TIL system.

- *Goals* are the purpose of agents' life and behaviour. An agent attempts to meet the goal assigned to them by applying particular rules of behaviour to their explicit knowledge stored in the knowledge base, and/or inferred by the inference machine.
- *Control* part executes the actions in accordance with a given agent's goal. In this way the agents perceive and influence their environment.

3 Knowledge Base

Knowledge base serves as agents' artificial memory. The content of the knowledge base consists of rules (obtained by communication, reasoning or domain ontologies) and facts (the state of the system and its environment).

3.1 Ontologies

Any content language is tightly related to ontologies. All concepts used or mentioned by a content language must be defined in agent's ontology. And *vice versa*, the content language must make it possible to use any concept from the ontology.

FIPA definition of ontology is relatively vague. It just says that ontology provides a vocabulary of domain specific concepts and relations between them. This leads to diversity in implementations. Actually, ontology takes a frame-like structure, which is well suitable for the FIPA SL language and developer frameworks like those supported by Jade.

The recent trend is to use well-estabiiehed technologies of semantic web, in particular the OWL language, for defining ontologies. However, the existing implementation tools for multi-agent systems do not support OWL in a sufficient way. The way we have chosen for TIL-Script is to inter-connect the language with the frame-like ontologies. This is due to the fact that TIL-Script is implemented in Jade. Integra-

tion of OWL into TIL-Script is a subject of our recent research. In OWL, ontology concepts (or classes) are sets of so-called individuals. We must not confuse these OWL-individuals with the members of the TIL-Script type Indiv. The correspondence between OWL-individuals and TIL-Script types is briefly this. Ontology OWL-individuals can be objects of any TIL-Script type α. This means that any ontology concept (class), the members of which are of a type α, is itself an object of type $(o\alpha)$, i.e. the set of α-objects. Ontology individuals (members of classes) are then α-objects.

Inter-connection of TIL-Script with ontology is mediated by the Trivialization construction, which is the primitive concept of a given entity X. Notational means in TIL is ${}^0X$, which corresponds to 'X in TIL-Script. You may Trivialize any object or individual defined by the ontology. The only demand for ontology to be used together with TIL-Script is that any class must have defined the TIL-Script type of its members. More details on the TIL-Script language can be found in (2007) and Ciprich – Duží – Košinár (2008).

Details on how rules and facts are stored, reconstructed and managed in Knowledge Base are described in next section.

4 Knowledge Management

Agent's knowledge base (in contrast to a human memory) is not a subject to forgettery in its native state. However, the amount of knowledge is limited by a memory capacity. Moreover, too much knowledge could lead to slow decision making and knowledge overwhelming. Thus it is necessary to save only those pieces of knowledge that are important from an agent's point of view, and discard the needless ones.

If a collection of agent's knowledge is too large and a need of memory clearing arises, it is necessary to determine those pieces of knowledge that most probably will not be needed by an agent any more. To this end it is possible and presumably suitable to apply some fuzzy rules, like, for example

- If a piece of knowledge is *new* then it is *important*
- If a piece of knowledge is *frequently used* and has been *recently used* then it is *important*

- If a piece of knowledge is *easily available* then it is not *important*

May seem to be some rules in dispute, but we must not forget that we work with the *fuzzy* rules, so the result depends on the *measure* of the newness, frequency, availability and the used aggregation function. For example the *very* new and *quite* easily available peace of knowledge can be determined as important.

Knowledge contraction can be performed not only in case of emergency, i.e. in the situation of a knowledge base overflow, but also for optimizing reasons. Thus the state of knowledge-base saturation is not the only trigger of knowledge contraction. This procedure should be executed regularly in order to speed up decision making so that no inefficient delays are caused by browsing through the useless knowledge records. Thus the fundamental rule guarding knowledge management can be formulated as follows.

Whenever a knowledge base is *too large* and contains some *unimportant* knowledge, then execute the process of knowledge contraction.

4.1 Knowledge model

Now we are going to describe the concrete model of knowledge representation and the way of its storing in the knowledge base. We will use three basic models of knowledge representation.

4.1.1 Basic Knowledge Relation – BKR

This relation model is based only on the content necessary for communication and reasoning of agents. It does not contain any attributes serving for optimization purposes. It is composed of the following parts.

- Content of data (can be divided into several parts: head, arguments and body, both for rules and facts)
- Knowledge encoding language (e.g. Prolog, TIL-Script, etc.)
- Time of knowledge obtaining

This model is simple and serves only for the purpose of testing; it is a predecessor of another model called *Extended Knowledge Relation*.

4.1.2 Extended Knowledge Relation – EKR

This relation model extends the *Basic Knowledge Relation* with other attributes that make it possible to apply the rules of knowledge management which we have sketched above.

- Content of data (can be divided into several parts: head, arguments, body)
- Knowledge encoding language (e.g. Prolog, TIL-Script, etc.)
- Time of knowledge obtaining
- Time of the last knowledge usage
- Frequency of using
- Origin (another agent, reasoning, original knowledge from basic ontology)
- Availability (this attribute can be represented by duration of regain of this knowledge from another agent)

4.1.3 Enhanced Extended Knowledge Relatione – EKR

EKR model contains enough information in order to manage knowledge in a reasonable way. Yet for optimization reasons, some more attributes proved to be useful. They are, for instance

- Place (subsystem) of knowledge obtaining
- Last place (subsystem) of knowledge usage
- Knowledge source (unique identification of knowledge source)

4.2 Erasing Knowledge

First of all, we must deal with so called availability problem. Assume that an agent A_1 obtained a piece X of knowledge from an agent A_2, and A_1 can communicate with A_2 anytime. Thus it might seem that A_1 does not have to remember the piece of knowledge X, because it is easily available from the agent A_2. But it is also possible that the agent A_2 is in the similar situation; it can regain this knowledge from the agent A_3; and so on (illustration on fig. 2). At the end of this sequence there might be an agent A_n who found the piece of knowledge X unimportant due to any reason specified in his knowledge contraction model. For example, it may be the case that X is not needed in the support of A_n's current goal. If this agent removes knowledge X, the

large queue of agents relying on this agent's knowledge get into troubles (fig. 3).

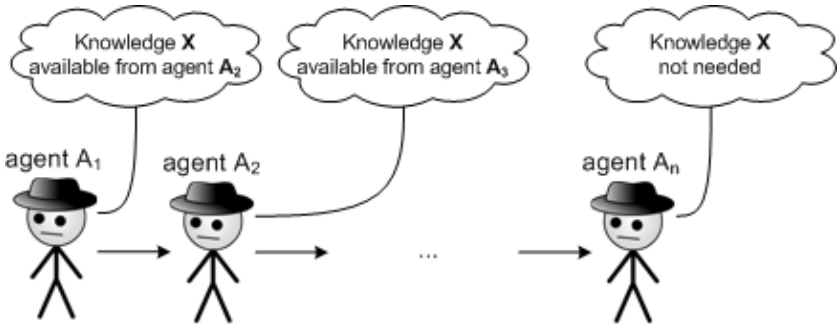

Figure 2. Availability problem - cause

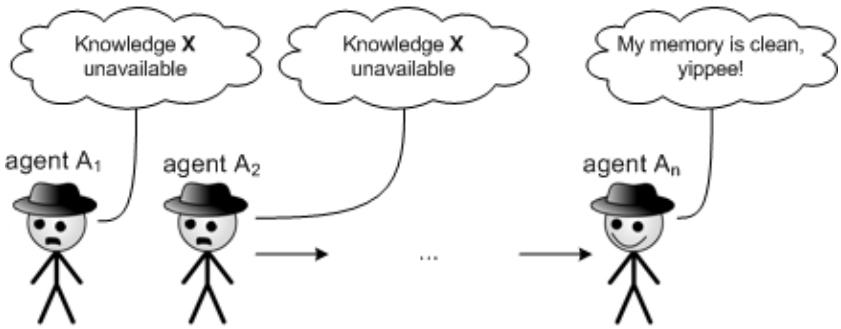

Figure 3. Availability problem - consequence

Thus we need to establish some knowledge contraction regularities. It is risky to design an agent in such a way that agent's behaviour is based on an arbitrary knowledge that might be stored only temporarily. Thus only "officially registered knowledge", typically knowledge registered at the Directory Facilities (original knowledge copied from ontology to agents' knowledge bases), is specified as reliable and safe. If an agent registered its facilities, then we can assume that these facilities or knowledge will not vanish.

Special care and attention must be paid to those knowledge rules that were obtained by reasoning and thus deduced from explicitly known facts, because it may be the case that this knowledge is unique in the whole system.

4.2.1 TTL - Time to Live

The Time to live algorithm implemented in knowledge management is much the same as in other implementations of this algorithm. It extends the knowledge relation with two other attributes; one of them is the limit of the knowledge lifetime (time to live constant) and the other contains a *current TTL* value.

When an agent obtains some knowledge, it is accompanied with the *TTL constant* (which is defined in realization phase by the programmer or knowledge base designer). When agent uses knowledge, the current value is set to the value of the *TTL constant*. When as agent checks the knowledge base with clearing procedure, all knowledge with zero value of *actual TTL* is removed. All the other *actual TTL* values are decremented by one.

Benefits – easy to implement, simple to understand

Disadvantages – are the latest unused knowledge really candidates to omit? The answer is no. This algorithm cleans all pieces of knowledge that were not used since the execution of the latest cleaning procedures (the count depends on the value of the *TTL constant*); however, this is not an optimal strategy of cleaning.

Usage – *TTL* omitting method is optimal for applications supporting real-time systems where are novel knowledge and the usage of old knowledge are quite rare.

4.2.2 TTL with Priorities

This cleaning method is much like the basic *TTL* we have just described. The extension is based on three additional attributes (all of them defined by a programmer or knowledge base designer).

One is the *original priority* of knowledge (i.e. *Very High*, *High*, *Middle*, *Low, …*) and is stored within the knowledge relation. The other is the actual priority value (this is set by the cleaning proce-

dure). The last one is the priority constant which contains information about priority values. This attribute should be stored in the system settings to ensure consistency of knowledge cleaning in whole system.

Desired values (i.e. *Very High=5, High=4, …, Low=2*) with the meaning 'how many times should the cleaning procedure waits till removing a given piece of knowledge'.

If a given piece of knowledge has been used in the interval between the execution of two cleaning procedures and the *current priority value* differs from the *original priority*, then the current value is replaced by the original one.

Example.

Rule *A*, *TTL Constant* = 3, *Original Priority* = *Middle*; *Priorities* {*Middle = 2, Low = 1*}
If *Knowledge A* is not used in the interval between the execution of three cleaning procedures then the cleaning procedure checks the *current priority value*; if the value is *Low* it will remove the knowledge; else if the value is *Middle*, it will check what is the subordinate value to "*Middle*" (currently it is *Low*) and will change the priority value to *Low*.

Benefits – same as of simple *TTL*

Disadvantages – same as of simple *TTL*

Usage – *TTL* method with priorities is also suitable for real-time systems but has some advantages that can facilitate the work with knowledge that should be more persistent than real-time knowledge. So the persistence is the main pro of *TTL with priorities* approach. Using priorities in knowledge management literally means that unneeded knowledge will be deleted but before this happens it will be considered more time to delete it based on the knowledge priority.

4.2.3 A priori Probability

Any knowledge relation that contains usage frequencies allows us to analyze unneeded entities *via* probability of knowledge usage. The idea is based on the fact that we can count the sum of all knowledge

usage frequencies. Then the probability of knowledge usage is defined by the following equality

$$P(Knowledge\ A\ Usage) = \frac{Frequency\ \ (Knowledge\ A)}{\sum_{i=1}^{n} Frequency\ \ (Knowledge_i)}$$

So far so good; we have the probability value of *Knowledge A* usage. Now we must obtain the value of the probability that *Knowledge A* will not be used. This can count as a complement of the probability of *Knowledge A* usage.

$$notP(Knowledge\ A\ Usage) = 1 - P(Knowledge\ A)$$

Now we have simple probability value of the fact that *Knowledge A* will not be used in the next phase between two cleaning procedures. A programmer, knowledge base designer or knowledge analyst must set the minimal probability limit that will define the level of knowledge removal.[1] When the value *notP* exceeds the limit value, this knowledge will be removed.

Benefits – the knowledge removal is based on a more precise algorithm than TTL. Other benefit is the statistical analysis of knowledge usage.

Disadvantages – there can be a problem of a new knowledge piece that has not been used till now, or has been used rarely in comparison to the other frequently used pieces of knowledge data. This problem can be solved away by limiting minimal knowledge usage (bad approach, because an agent may never use it) or limiting the minimal time of knowledge ownership. Another way of avoiding probability problem is using the *Conditional Probability* or a combination of simple *Probability* with *TTL method*. Both are described below.

[1] Minimal value of probability *notP*.

Usage – probability algorithm should be used in general multi-agent systems where the knowledge usage is more or less regular.

4.2.4 Conditional Probability

We can avoid many inaccuracies in knowledge cleaning by using conditional probability instead of simple probability of knowledge usage. However, conditional probability has one limitation, namely the obligatory usage of *Enhanced Extended Knowledge Relation* (*eEKR*). This need arises from the fact that the conditional probability of knowledge (non-) usage must be combined with the condition probability.

The conditions are taken from additional attributes of eEKR and can be based on varying information (see 0 for more details).

$$P(Knowledge\ A\ notUsage \mid Conditions) = \frac{P(KnowledgeAnotUsage \cap Condition)}{P(Conditions)}$$

We can optimize the cleaning procedure with many domain (or application) conditions like knowledge obtain place, actual place, knowledge ownership duration, probabilities of knowledge usage from other agents and so on.

Benefits – the knowledge removal is based on even more precise algorithm than simple Probability. Other benefit is statistical analysis of knowledge usage. The added value is an enrichment of probability knowledge cleaning with domain specific information which increases cleaning reliability.

Disadvantages – as in simple probability, situations where knowledge were used sporadically compared to other pieces of knowledge can create a problem. The very same is the solution of the problem – limiting the minimal time of knowledge ownership or combining with *TTL* method.

Usage – probability algorithm should be used in general multi-agent systems where the knowledge usage is more or less regular.

4.2.5 Combinations of TTL and Probabilities methods

Knowledge management algorithms can be combined in order to gross up the hit rate of searching useless knowledge. Combination of both methods means that both algorithms must mark a piece of knowledge as unneeded to allow the cleaning procedure to remove it.

4.3 C-LURF Management Model

The C-LURF notion stands for the formal representation of knowledge management model we are using in our applications. The acronym 'C-LURF' means Communication, Learning, Reasoning, Using and Forgetting where the order of the words is equivalent to the order of the basic knowledge management.

As the knowledge management is much more complicated in real applications, we prefer more transparent graphical representation of the C-LURF model which is illustrated by Figure 4 below.

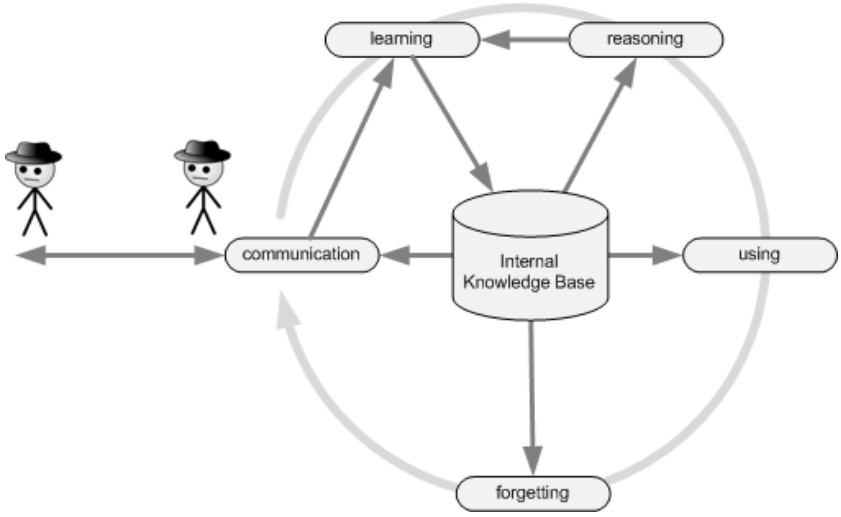

Figure 4. C-LURF Management Model: Dark arrows represent the knowledge flow.

In the model there are all knowledge flows that may occur in an agent's lifetime. The basic light grey circle arrow is the very C-LURF flow. Dark grey arrows stand for:

- knowledge base → communication – flow that prepares knowledge necessary for communication among agents. The prerequisite is that necessary knowledge must exist in knowledge base.
- communication → other agents – flow that represents knowledge exchanged between agents. For incoming flows there are no prerequisites. For outgoing flows the prerequisite is the knowledge preparation.
- communication → learning – this flow actually represents just the fact that every agent can use communication for learning new rules (or facts).
- learning → knowledge base – there would be no sense of learning without the ability of memorizing its results. This data flow allows the agent to store a novel piece of knowledge.
- knowledge base → reasoning – Every agent must be autonomous and reasonable, so reasoning is its basic property. However, in order to deduce something, the basic knowledge for the procedure must be provided and the source is knowledge base.
- reasoning → learning – if the reasoning procedure has some results, then we can say that an agent has learnt something new.
- knowledge base → using – by using knowledge we mean every flow that comes out from knowledge base. The "using" part of the model is an abstraction of all knowledge usage in this model.
- knowledge base → forgetting – when knowledge gets through this management procedure it is forever lost from the knowledge base – it is forgotten.

5 Conclusion

The knowledge base design and knowledge management is still work in progress. In this paper we introduced and described knowl-

edge management in general. We have illustrated the process of building knowledge bases for multi-agent systems. We analyzed a model of knowledge representation and proposed some methods for knowledge omitting purposes using basic statistical approaches like *Time To Live* algorithm and its advanced version *TTL with Priorities* and usage of *Simple* and *Conditional Probabilities*.

We have also introduced the knowledge flow model called *C-LURF* which formally describes our implementations of knowledge based multi-agent systems. This model can bring closer multi-agent system communication and learning to knowledge base analysts and designers.

Acknowledgment

This research has been supported by internal grant agency of FEECS VSB-TU Ostrava – IGA 22/2009 Modeling, simulation and verification of software processes. Author is also supported as a *Grand aided student of Municipality of Ostrava, Czech Republic*.

VŠB – Technical University Ostrava
Faculty of Electrical Engineering and Computer Science
17. listopadu 15
708 33 Ostrava
Czech Republic
michal.kosinar@vsb.cz, ondrej.kohut.fei@vsb.cz

References

CIPRICH, N. – DUŽÍ, M. – KOŠINÁR, M. (2007): TIL-Script: Functional Programming Based on Transparent Intensional Logic. In: Sojka, P. - Horák, A. (eds.): *RASLAN* (2007). Brno: Masaryk University, 37 – 42.

CIPRICH, N. – DUŽÍ, M. – KOŠINÁR, M. (2008): *The TIL-Script language*. In the Proceedings of the 18th European Japanese Conference on Information Modelling and Knowledge Bases (EJC 2008), Tsukuba, Japan 2008.

Foundation for Intelligent Physical Agents. FIPA Abstract Architecture Specification. http://fipa.org/specs/fipa00001/, 2000a.

Foundation for Intelligent Physical Agents. FIPA SL Content Language Specification. http://fipa.org/specs/fipa00008/, 2000b.

LUCK, M. – MCBURNEY, P. – SHEHORY, O. – WILLMOTT, S. (2005): Agent Technology: Computing as Interaction. A Roadmap for Agent-Based Computing. *AgentLink* (2005).